NAME

bios_sigsearch, **bios32_SDlookup**, **bios32**, **bios_oem_strings** - interact with PC BIOS

SYNOPSIS

```
#include <sys/param.h>
#include <vm/vm.h>
#include <vm/pmap.h>
#include <machine/pc/bios.h>

uint32_t
bios_sigsearch(uint32_t start, u_char *sig, int siglen, int paralen, int sigofs);

int
bios32_SDlookup(struct bios32_SDentry *ent);

int
bios32(struct bios_regs *br, u_int offset, u_short segment);

BIOS_PADDRTOVADDR(addr);

BIOS_VADDRTOPADDR(addr);

extern struct bios32_SDentry PCIbios;
extern struct SMBIOS_table SMBIOStable;
extern struct DMI_table DMItable;

int
bios_oem_strings(struct bios_oem *oem, u_char *buffer, size_t maxlen);

struct bios_oem_signature {
        char * anchor;          /* search anchor string in BIOS memory */
        size_t offset;          /* offset from anchor (may be negative) */
        size_t totlen;          /* total length of BIOS string to copy */
};
struct bios_oem_range {
        u_int from;             /* shouldn't be below 0xe0000 */
        u_int to;               /* shouldn't be above 0xfffff */
};
struct bios_oem {
        struct bios_oem_range range;
```

```
        struct bios_oem_signature signature[];
};
```

DESCRIPTION

These functions provide a general-purpose interface for dealing with the BIOS functions and data encountered on x86 PC-architecture systems.

bios_sigsearch() Searches the BIOS address space for a service signature, usually an uppercase ASCII sequence surrounded by underscores. The search begins at *start*, or at the beginning of the BIOS if *start* is zero. *siglen* bytes of the BIOS image and *sig* are compared at *sigofs* bytes offset from the current location. If no match is found, the current location is incremented by *paralen* bytes and the search repeated. If the signature is found, its effective physical address is returned. If no signature is found, zero is returned.

bios_oem_strings() Searches a given BIOS memory range for one or more strings, and composes a printable concatenation of those found. The routine expects a structure describing the BIOS address *range* (within 0xe0000 - 0xfffff), and a { NULL, 0, 0 } -terminated array of *bios_oem_signature* structures which define the *anchor* string, an *offset* from the beginning of the match (which may be negative), and *totlen* number of bytes to be collected from BIOS memory starting at that offset. Unmatched anchors are ignored, whereas matches are copied from BIOS memory starting at their corresponding *offset* with unprintable characters being replaced with space, and consecutive spaces being suppressed. This composed string is stored in *buffer* up to the given *maxlen* bytes (including trailing '\0', and any trailing space suppressed). If an error is encountered, i.e. trying to read out of said BIOS range, other invalid input, or *buffer* overflow, a negative integer is returned, otherwise the length of the composed string is returned. In particular, a return value of 0 means that none of the given anchor strings were found in the specified BIOS memory range.

BIOS_VADDRTOPADDR()
Returns the effective physical address which corresponds to the kernel virtual address *addr*.

BIOS_PADDRTOVADDR()
Returns the kernel virtual address which corresponds to the effective physical address *addr*.

SMBIOStable — If not NULL, points to a *struct SMBIOS_table* structure containing information read from the System Management BIOS table during system startup.

DMItable — If not NULL, points to a *struct DMI_table* structure containing information read from the Desktop Management Interface parameter table during system startup.

BIOS32

At system startup, the BIOS is scanned for the BIOS32 Service Directory (part of the PCI specification), and the existence of the directory is recorded. This can then be used to locate other services.

bios32_SDlookup() — Attempts to locate the BIOS32 service matching the 4-byte identifier passed in the *ident* field of the *ent* argument.

bios32() — Calls a bios32 function. This presumes that the function is capable of working within the kernel segment (normally the case). The virtual address of the entrypoint is supplied in *entry* and the register arguments to the function are supplied in *args*.

PCIbios — If not NULL, points to a *struct bios32_SDentry* structure describing the PCI BIOS entrypoint which was found during system startup.